Taming Cryptography
by aertsimon90
(Turkish Translate)

Bölüm 1: Keşfin Kıvılcımı

Kriptografi dünyasına yolculuğum, saf bir cehaletin kollarında başladı. O zamanlar şifreleme ya da onun karmaşıklığıyla hiçbir bağım yoktu. Benim için veriyi güvence altına almak, bir sunucuya kaydetmek ve kimsenin kıramayacağını sandığım dijital bir kilidin ardına saklamak kadar basitti. Bu özgüvenim, sunucunun güvenli olduğu sürece verinin dokunulmaz olduğuna dair naif bir inançtan geliyordu. Ama yanılmıştım—hem de fena halde. Asıl zayıflık, sunucuya erişim değil, internetin damarlarında akan, savunmasız ve açıkta olan veriydi. Bu gerçek, bir yıldırım gibi çarptı ve her şeyi değiştirdi.

Web'in motoru olan HTTP protokolü hakkında biraz bilgim vardı. HTTP üzerinden gönderilen verinin şifrelenmediğini, doğru araçlara sahip herkesin okuyabileceği bir düz metin olarak seyahat ettiğini biliyordum. Güvenli kuzeni HTTPS, o zamanlar bana gereksiz bir lüks gibi geliyordu. Sunucum "güvenli" olduğuna göre neden zahmet edeyim ki? Ama sonra utandırıcı bir gerçeği öğrendim: HTTPS bile, ortadaki adam saldırıları ya da sertifika sahteciliği gibi sofistike yöntemlerle kırılabiliyordu. Birdenbire HTTPS, sadece bir lüks değil, modern güvenliğin temel taşı haline geldi—özellikle benim gibi, kriptografinin uçsuz bucaksız, kaotik dünyasına yeni adım atan bir çaylak için. Yıl 2023'tü ve veri, güvenlik, bilgiyi gizleme sanatı üzerine algımı yeniden şekillendirecek bir maceraya atılmak üzereydim.

Şifrelemeyle ilk tanışmam, en basit algoritmalardan biriyle oldu: Sezar Şifresi. Bu algoritma, sadeliğiyle beni büyüledi—Julius Caesar'ın kendisi mesajlarını korumak için kullanmıştı. Alfabe harflerini sabit bir sayıda kaydırma fikri, casus romanlarından fırlamış bir gizli kod gibiydi. Ama acemi bir coder olarak, bu basit algoritma bile gözümü korkutuyordu. Seçtiğim programlama dili Python'a sarıldım ve ChatGPT'nin biraz yardımıyla ilk Sezar Şifresi kodumu yazmayı başardım. Bu bir zaferdi, ama aynı zamanda bir bilmece. Kod, anlamadığım kavramlarla doluydu; mesela modül operatörü (%). Bunun modüler aritmetik için kullanıldığını, sayıları sabit bir aralıkta döndürdüğünü bilmiyordum. Benim için sadece şifrenin çalışmasını sağlayan gizemli bir semboldü.

Sezar Şifresi'yle erken deneylerim hem heyecan verici hem de sinir bozucuydu. Basit bir mesajla, "Merhaba" ile test ettim ve harflerin 3'lük bir kaymayla "Pıuod" gibi bir şeye dönüştüğünü izledim. Sihir gibiydi. Ama sonra bir sınırlama fark ettim: Şifre, sadece alfabetik karakterlerde—özellikle Latin harflerinde—çalışıyordu. 29 karakterli bir alfabe (A-Z artı birkaç eklediğim karakter) ile sayılar, semboller ve boşluklar görmezden geliniyordu. Farklı anahtarlarla oynadım: 2'lik kayma, sonra 7, ve bir hırs patlamasıyla devasa bir 372'lik kayma. Şaşırtıcı şekilde, 372'lik kayma, 24'lük kaymayla aynı sonucu verdi. Neden? Kafamı kaşıdım, afallamıştım. Farklı bir anahtar nasıl aynı sonucu üretebilirdi? Sanki şifre benimle dalga geçiyordu.

Cevap, Sezar Şifresi'nin alfabesinin döngüsel olduğunu fark ettiğimde geldi. Sadece 29 karakterle, 29'un ötesindeki her kayma geri dönerdi. 372'lik bir kayma, 372 % 29 = 24'e eşitti. Bu, modüler aritmetiğin ilk dersiydi, ama o zamanlar bunun adını bilmiyordum. Ayrıca, 29 karakterli bir alfabe ile sadece 29 olası anahtar olduğunu keşfettim—şok edici derecede

küçük bir sayı. O zamanlar bu yeterli görünüyordu. “Kim 29 anahtardan birini tahmin edebilir ki?” diye düşünmüştüm. Şimdi geriye bakınca, ne kadar saf olduğuma gülüyorum. Yirmi dokuz anahtar? Bu, kriptografik bir şaka, saniyeler içinde kaba kuvvetle kırılır.

Yılmadan, Sezar Şifresi’ni daha derinlemesine inceledim, ince ayar yaptım, denemeler yaptım. Onu TT3 adını verdiğim küçük bir hack aracıma entegre ettim; bu, yeni şifreleme fikirlerimi test ettiğim kişisel bir projeydi. Şifrelere, hatta kendi şifrelerimi yaratma fikrine âşık olmuştum. Daha hırslı algoritmalar hayal etmeye başladım, her biri bir öncekinden biraz daha iddialı. En gurur duyduğum buluşlardan biri, metni görünmez Unicode karakterlerine dönüştüren bir sistemdi—gözle görülmeyen, gizemli karakterler. Sanki kadim bir kodu kırmış, mesajları dijital hayaletlere dönüştürmüştüm. İlk algoritmalarımı gurur ve biraz da muziplikle isimlendirdim:
- Simon UNI Encrypter (Unicode Parşömeni): Unicode’un geniş karakter setini kullanan bir şifre.
- SimonOperator Encrypter: Daha karmaşık, kaydırma tabanlı bir sistem denemem.
- Simon WayOne Encrypter: Tek yönlü bir kodlama deneyi.
- Enblank/Deblank: Metni görünmez Unicode karakterlerine çeviren ilginç bir araç.

Bunlar, kriptografiye ilk adımlarımdı—kaba ama coşku doluydu. Bu bölümü yazarken, eski bir yaratımıma geri dönmeye karar verdim: SimonOperator Encrypter, bir zamanlar başyapıtım saydığım sistem. Basit bir giriş, “Merhaba!” verdim ve görünüşte rastgele Unicode karakterlerinden oluşan bir dizi çıkardığını izledim: □□□□□□. 2023’te, Unicode’un tam 1,114,112 karakterlik yelpazesine takıntılıydım. Çıktı kaotik görünüyordu, başka bir boyuttan gelen bir kriptografik eser gibi. Kırılmaz bir şey yarattığımı sanıyordum. Ama henüz görmediğim bir kusur vardı, yıllarca peşimi bırakmayacak bir kusur.

O zamanlar, Sezar Şifresi’nden kalma bir alışkanlıkla sayısal anahtarlar kullanıyordum. “Karmaşık” SimonOperator Encrypter’ımın sadece süslü bir kaydırma şifresi olduğunu, her Unicode karakteri için bir anahtar olmak üzere 1,114,112 olası anahtarla sınırlı olduğunu fark etmemiştim. O günlerde bu sayı devasa geliyordu. Bir milyon anahtar? Kırılmaz, demiştim. Şimdi, kendi cehaletime gülüyorum. Modern kriptografide 1,114,112 anahtar gülünç derecede az, kaba kuvvet saldırılarına anında yenik düşer. Kumdan kaleler inşa ediyordum, yaklaşan dalgaların farkında değildim.

2024 ve 2025’e doğru ilerlerken, deneylerimle birlikte özgüvenim de büyüdü. ShieldKEY adında yeni bir algoritma geliştirdim, bunun yenilmez olduğuna inanıyordum. Saatlerimi buna harcadım, bunun nihai şifreleme sistemi olduğundan emindim. Ama aynı kusur devam ediyordu: ShieldKEY, önceki yaratılarım gibi, Unicode kaydırmalarının 1,114,112 anahtar sınırına zincirlenmişti. Bu zayıflığı göremiyordum, yaratmanın heyecanı varsayımlarımı sorgulamama engel oluyordu. Sistemler inşa ettim, projeler test ettim, şifrelerimi gerçek dünya uygulamalarına entegre etme hayalleri kurdum. Ama içimi kemiren bir şüphe vardı, kaybolmayan bir huzursuzluk.

Algoritmalarımın ne kadar “güvenli” olduğunu düşünsem de, onları projelerime entegre etmekte zorlanıyordum. İzole bir şekilde çalışıyorlardı, ama gerçek sistemlere—mesela veri aktarımını güvence altına almak ya da bir veritabanını korumak—uygulamaya çalıştığımda yetersiz kalıyorlardı. Kod hantaldı, güvenlik kanıtlanmamıştı, performans yavaştı. Oyuncaklar yapmıştım, araçlar değil. Şifrelerim, bana etkileyici gelen ama gerçek dünya zorlukları

karşısında işe yaramaz parlak biblolardı. Temel bir şeyi kaçırıyordum, amatör deneylerimi interneti güçlendiren sağlam sistemlerden ayıran bir şeyi.

Bu şüphe, bir dönüm noktası oldu. Karanlıkta oyalanarak, kriptografinin derin prensiplerini anlamadan algoritmalar yamalamaya devam edemeyeceğimi fark ettim. Ciddi bir şey inşa etmeliydim, eleştirilere dayanabilecek, kendi tatminimden öte bir amaca hizmet edebilecek bir şey. Hızlı, güvenli ve modern veri güvenliğinin karmaşıklıklarını kaldırabilecek çok yönlü bir şifreleme motoruna ihtiyacım vardı. İşte o zaman *LeCatchu* fikri doğdu.

LeCatchu sadece bir algoritma değildi; bir vizyondu. Şifrelemenin kaosunu dizginleyebilecek, aynı zamanda kontrollü ve güvenilir kalabilecek bir motor hayal ettim. Bu sistem, Sezar Şifresi'nin sadeliğinden ShieldKEY'in kibrine kadar hatalarımdan ders çıkaracaktı. Devasa anahtar uzaylarını destekleyecek, kaba kuvvet saldırılarına direnecek ve gerçek dünya uygulamalarına sorunsuzca entegre olacaktı. En önemlisi, sadece bir konsept değil, güvenebileceğim bir araç olacaktı. Keşfin kıvılcımı bir yangına dönüşmüştü ve ben kriptografiyi evcilleştirmeye—ya da en azından denemeye—hazırdım.

Geriye bakınca, o ilk günler alçaltıcıydı. Temelleri tökezleyerek öğrenen, modül işlemini yanlış anlayan ve 29 anahtarlı şifreleri Fort Knox sanan bir çaylaktım. Ama her yanlış adım bana bir şey öğretti. Sezar Şifresi bana sadeliğin güzelliğini gösterdi. SimonOperator Encrypter, Unicode'un gücünü ve sınırlarını ortaya koydu. ShieldKEY, kendi kibrimle yüzleşmemi sağladı. Ve o içimi kemiren şüphe, beni *LeCatchu*'ya itti. Kriptografi, sadece veriyi gizlemek değildi; kaosu anlamak, ona hâkim olmak ve dijital dünyanın fırtınalarına dayanabilecek sistemler inşa etmekti.

Bu sadece başlangıçtı. *LeCatchu*'ya giden yol uzun olacaktı, denemeler, atılımlar ve birkaç başarısızlıkla dolu. Ama 2023'te, ChatGPT'nin yardımıyla ilk Sezar Şifresi'ni yazarken, çok daha büyük bir şeyin tohumlarını ektiğimin farkında değildim. Öğreniyordum—sadece şifrelemeyi değil, keşfi körükleyen durdurulamaz merakı da. Ve bu merak, beni adım adım ileriye taşıyacaktı, bir şifreleme zaferi peşinde.

Bölüm 2: LeCatchu'nun Doğuşu

LeCatchu'nun tohumu, bir hayal kırıklığı ve vahiy anında ekildi, ama kökleri, ilk kriptografik tökezlemelerime kadar uzanıyordu. Önceki projelerimdeki—ShieldKEY, EnigmaDB ve sadece 1,114,112 benzersiz anahtarla sınırlı bir dizi Unicode tabanlı şifre—ölümcül kusuru fark ettikten sonra, farklı bir şeye ihtiyacım olduğunu biliyordum. Cesur bir şey. Geçmişimin sınırlarını kırıp şifreleme anlayışımı yeniden tanımlayabilecek bir şey. İşte o şey *LeCatchu*'ydu, aniden aklıma gelen, kriptografik destanımı tanımlayacak bir isim.

LeCatchu isminin büyük bir köken hikayesi yoktu. Antik mitolojiden ya da bir deha anından doğmadı. Dürüst olmak gerekirse, sadece uydurdum, kodlama sığınağımda—kendi tabirimle "tapınağımda"—otururken aklıma geldi. Kulağa havalı, belki biraz gizemli geliyordu. Biraz çekicilik katmak için önüne "Le" ekledim, İtalyan şarabı ya da Rönesans tablosu havası versin diye. *LeCatchu*. Dilimde yuvarlanıyordu, hiçbir anlamı olmasa bile. Sonra değiştiririm dedim, ama içten içe mükemmel olduğunu biliyordum. Özünde bir Türk projesi olan *LeCatchu*, sıradanlığa karşı isyanımdı, sıfırdan olağanüstü bir şey inşa edebileceğime dair bir beyan.

O ilk günlerde, tek başıma bir koderdim, merak ve inatçı bir vazgeçmeme azmiyle doluyordum. Arkadaşlarım ara sıra projelerimi test ederdi, geri bildirim verirdi ya da sadece takıntımı hoş görürdü, ama *LeCatchu* benim bebeğimdi. Yalnız çalıştım, klavyemin başında kambur, yaratmanın heyecanını kovalayarak. Dış dünya soluklaşırken, Python satırları ekranımı doldurdu, her biri daha büyük bir şeye doğru bir adımdı. Ama geçmiş hatalarımın hayaletleri peşimi bırakmıyordu. 1,114,112 anahtar sınırı sadece teknik bir kusur değildi; kişisel bir başarısızlıktı. EnigmaDB gibi uzun zaman önce terk ettiğim bir veritabanı sistemi ve "kırılmaz" dediğim ShieldKEY, aynı sallantılı temeller üzerine inşa edilmişti. Hepsinin savunmasız olduğunu, kaba kuvvetle saniyeler içinde kırılabileceğini fark ettiğimde, ayaklarımın altındaki zemin çökmüş gibi hissettim. EnigmaDB mi? Belki bu yazıyı bitirdikten sonra silerim, dedim. ShieldKEY mi? Cehaletimin bir anıtı. Daha iyisini yapmalıydım.

LeCatchu'nun ilk versiyonu, bu yüzleşmeden doğdu. Kalbimi buna adadım, anahtar uzayı sorununu çözmeye kararlıydım. Ama kolay olmadı. İlk denemelerim karmakarışıktı, işbirliği yapmayı reddeden kaotik bir kod yığınıydı. Her şeyi denedim—algoritmaları ayarladım, fonksiyonları yeniden yazdım, yeni modellerle oynadım—ama 1,114,112 anahtar sınırı bir lanet gibi üzerimdeydi. Her başarısızlık canımı yaktı, stres beni yıpratmaya başladı. Bir gece, öfkeden telefonumu odaya fırlattım ve yatağa yığıldım. Bir iki gün boyunca hiçbir şey yapmadım. Ne kod yazdım, ne düşündüm, sadece tamamen koptum. Tam da ihtiyacım olan buydu—kendi takıntımın sisini dağıtmak için bir sıfırlama.

LeCatchu'ya geri döndüğümde, bir ilham kıvılcımı çaktı. Basit, neredeyse gülünç derecede basit, ama her şeyi değiştirdi. Sorun sadece anahtar sayısı değildi; şifreleme sürecinin onları nasıl ele aldığıydı. Eski şifrelerimde, benzer girişler benzer çıktılar üretiyordu çünkü kaydırma (ya da anahtar) sabitti. 3 anahtarlı bir Sezar Şifresi, giriş ne olursa olsun harfleri 3 kaydırırdı. Unicode tabanlı şifrelerim bile aynı mantığı izliyordu, sadece daha büyük bir alfabe ile. Ama ya anahtarın kendisi her karakterle değişseydi? Ya her karakteri şifreledikten sonra algoritma, sonraki kaydırmayı benzersiz kılacak şekilde anahtarı dönüştürseydi? İşte bu bir devrimdi: dinamik, kendi kendine evrilen bir anahtar.

Fikir, her karakterden sonra anahtarı hash'lemek ve bir sonrakine yeni bir anahtar üretmekti. Böylece, aynı girişler bile tamamen farklı çıktılar üretecekti, çünkü anahtar sürekli değişiyor, verinin kendisi tarafından şekilleniyordu. Şifre, kendini şifreliyordu, katman katman, öngörülemezlik kaskadı yaratıyordu. Hemen işe koyuldum, bu teoriyi test etmek için çılgınca kod yazdım. *LeCatchu*'nun ilk modeli hamdi, ama umut vadediyordu. Özel bir hash fonksiyonu kullandım, anahtarı her karakterden sonra dönüştürmek için bir üs alma işlemi (mesela 2.38382947'lik bir üs) yaptım. Bir an için çözdüğümü sandım. Çıktılar benzersizdi, anahtar uzayı sonsuz gibiydi ve ben sevinçten havalardaydım.

Ama sonra felaket. Özel hash fonksiyonumda ölümcül bir kusur vardı: anahtarın değerini sürekli büyütüyordu. Her karakterle sayı şişiyor, sonsuza yaklaşıyordu. Python'da sayılar teorik olarak sınırsız büyüyebilir, ama bir tuzak var: "inf" (sonsuz) değerine ulaştıklarında anlamlarını yitiriyorlar. Her sonsuz değer aynı, farkların kaybolduğu bir tekillik. Anahtarlar büyüdükçe bu sonsuz boşluğa yaklaştı ve şifrelenmiş çıktılar ürkütücü şekilde birbirine benzemeye başladı. Daha kötüsü, şifre tuhaf bir davranış sergilemeye başladı: *yanlış anahtarla bile* doğru çözülüyordu. Yakınsayan anahtarlar, tüm yolların aynı hedefe çıkması

demekti, şifrelemeyi işe yaramaz kılıyordu. Bu, kriptografi utanç salonuna girecek destansı bir başarısızlıktı. Kendini ihanet eden bir şifre yapmıştım.

Hüsran geri döndü, ve onunla birlikte bir mola. Uzaklaştım, ama *LeCatchu* aklımdan hiç çıkmadı. Çalıntı anlarda—telefonumda, yatakta, molalarda—testler yaptım, hatayı kovaladım ve tekrar tekrar başarısız oldum. Vazgeçmeye hazırdım, bir çıkmazda olduğuma inanıyordum. Ama sonra, o klişe anlardan birinde ilham geldi. Şunu hayal et: Tuvalette oturuyorum, telefonumda kaydırıyorum, ve fikir bir şimşek gibi çakıyor. “Neden anahtarı büyüten özel bir hash kullanıyorum ki? Neden SHA-256 kullanmayayım?” O kadar bariz, o kadar zarifti ki, yüksek sesle güldüm. SHA-256 hızlı, güvenli ve—en önemlisi—sabit uzunlukta çıktılar üretiyordu. Kaçak sayılar yok, sonsuzluk tuzakları yok, sadece tutarlı, çarpışmaya dirençli hash’ler.

Bilgisayarıma koştum, ChatGPT ve diğer yapay zeka araçlarından yardım alarak bu yeni yaklaşımı uyguladım. Hatalı hash fonksiyonumu SHA-256 ile değiştirdim, her karakterden sonra anahtarı dönüştürmek için kullandım. Sonuçlar inanılmazdı. Şifre kusursuz çalışıyordu, her giriş ve anahtar kombinasyonu için benzersiz çıktılar üretiyordu. Ayrıca SHA-256’nın onaltılık çıktısını tamsayılara dönüştürmeyi öğrendim, bu yeni olasılıklar açtı. Ama bir pürüz vardı: SHA-256’nın 64 karakterli hex dizileri nispeten kısaydı, bazı uç durumlarda anahtar çarpışmalarına yol açabilirdi. “Yeterince iyi” ile yetinmedim. Mükemmeliyet istiyordum. Vigintilyonlarla sınırlı olmayan, yazılım dünyasında sonsuza yaklaşan bir anahtar uzayı istiyordum.

İşte o zaman *xbase*’i tanıttım, *LeCatchu*’nun benzersizliğinin kalbi olacak kutsal bir değişken. *xbase*, hash çıktısının uzunluğunun çarpanıydı. *xbase=1* ile hex dizisi 64 karakterdi. *xbase=2* ile 128 karaktere çıkıyordu. Daha ileri gittim, *xbase*’i 40’a kadar ölçeklendirdim, hesaplama açısından pratikte sonsuz olan anahtarlar yarattım. Bu sadece bir düzeltme değildi; bir devrimdi. *LeCatchu*, eski şifrelerimi cüceleştiren bir anahtar uzayına sahipti, *xbase=1* ile vigintilyonlarca benzersiz anahtar ve daha yüksek değerlerde hayal edilemez permütasyonlar sunuyordu. Kaba kuvvet saldırıları mı? Gülünç. Çarpışmalar mı? Neredeyse imkânsız. Kriptografik bir kale inşa etmiştim.

Bu atılımla, *LeCatchu* v1 doğdu. Kaba, cilasız, ama çalışıyordu. Kutlamak için bir README.md dosyası hazırladım, heyecanımı her satıra döktüm. Teknik bir belgeden fazlası olsun istedim; *LeCatchu*’nun kaotik parlaklığının hikayesini anlatsın. README cesurdu, belki biraz dramatik, ama yarattığım şeyin ruhunu yakalıyordu. *LeCatchu* v1 bir dönüm noktasıydı, azmimin ve daha azına razı olmama inadımın kanıtıydı. Tek kelimeyle, muhteşemdi.

Ama bu sadece başlangıçtı. *LeCatchu* canlıydı ve daha fazlasını istiyordu. V1’in kusurları—verimsizlikleri, tuhaflıkları—zaten kulağıma fısıldıyordu, iyileştirme talep ediyordu. Önümdeki yolun uzun olduğunu, yeni zorluklar ve keşiflerle dolu olduğunu biliyordum. Ama şimdilik, tamamen bana ait, başarısızlıktan doğmuş ve kararlılıkla dövülmüş büyük bir şeyin eşiğindeydim. *LeCatchu* sadece bir şifreleme motoru değildi; sıradanlığa karşı isyanımdı, kriptografinin vahşi kaosunu evcilleştirme arayışımdı.

Bölüm 3: LeCatchu’yu Dövmek

LeCatchu'nun yaratılışı, önceki kriptografik deneylerimin kusurlarına karşı bir isyan, bir hüsran kıvılcımıydı. Ama bu kıvılcımı yanan bir şifreleme motoruna dönüştürmek, amansız kodlama, ezici başarısızlıklar ve zor kazanılan zaferlerle dolu zorlu bir yolculuktu. V1'in ham kaosundan v5'in cilalı parlaklığına, LeCatchu beş büyük iterasyonla evrildi, her biri kendi sınırlarım ve kriptografinin acımasız talepleriyle bir mücadeleydi. Bu bölüm, o evrimin hikayesi, her hataya, her atılıma ve LeCatchu'yu hesaba katılması gereken bir güce dönüştüren her derse derin bir dalış.

LeCatchu v1'i piyasaya sürdüğümde, bu vahşi, evcilleşmemiş bir canavardı, ShieldKEY ve EnigmaDB gibi önceki şifrelerimin ölümcül kusurunu aşmak için inşa edilmişti: 1,114,112 benzersiz anahtardan oluşan zavallı bir anahtar uzayı. Beni kurtaran vahiy, hash çıktılarının uzunluğunu katlayan kutsal bir parametre olan xbase'ti, bu sonsuz gibi hissettiren bir anahtar uzayı yarattı. SHA-256 kullanarak, her karakterden sonra anahtarı hash'ledim, her şifreleme adımını dinamik ve benzersiz kıldım. xbase=1 ile hex dizisi 64 karakterdi; xbase=40'ta 2560 karaktere uzanıyordu, vigintilyonlarca anahtar sunuyordu. Bu, tek bir kaydırmanın hükmettiği eski Unicode tabanlı şifrelerimden kuantum bir sıçrayıştı.

Ama v1'in kenarları pürüzlüydü. Hash sistemi, anahtarı kendisiyle birleştiren (key + okey) hantal bir yamaydı. Bu yavaştı, CPU döngülerini aç bir canavar gibi yiyordu. 1024 baytlık bir mesajı şifrelemek 0.0162 saniye sürüyordu, bir prototip için fena değil ama pratikten uzak. Şifreleme kendisi basitti, hash'lenmiş anahtara dayalı olarak karakterleri kaydırmak için modüler aritmetik kullanıyordu, steroidli bir Sezar Şifresi'nden farksızdı. Unicode'un tam 1,114,112 karakterlik yelpazesini destekliyordu, bu kulağa etkileyici gelse de, özellikle büyük metinlerde kodlama ve kod çözmeyi yavaşlatıyordu.

Sonra gizli bir kusur vardı: hash çarpışmaları. Nadir durumlarda, özellikle düşük xbase değerlerinde, farklı girişler aynı hash'i üretebiliyordu, bu da şifrelemeyi öngörülebilir kılıyordu. Bu, kriptografik bir günah, temelde saldırganların sızabileceği bir çatlaktı. Başka bir sorun veri işleme idi. LeCatchu v1 sadece dizileri şifreleyebiliyordu, listeler ya da sözlükler gibi karmaşık verileri dışarıda bırakıyordu. Şifrelemeden önce bunları elle seri hale getirmem gerekiyordu, amatörlüğü bağıran sıkıcı bir geçici çözüm. Bu kusurlara rağmen, v1 bir konsept kanıtıydı, verimli toprağa ekilmiş kaotik bir tohum. Heyecanımı gösterişli bir README.md'ye döktüm, LeCatchu'yu entropinin evcilleştiricisi ilan ettim ve daha iyisini yapma sözü verdim.

LeCatchu v2, v1'in kaosunu dizginlemeye yönelik ilk girişimimdi, ama ruhunu kaybetmeden. Hash sistemi öncelikliydi. Anahtar dönüşümünü kolaylaştırdım, birleştirmeleri azalttım ve SHA-256 çağrılarını optimize ettim. Bu, şifreleme süresini 1024 bayt için 0.0162 saniyeden 0.0152 saniyeye indirdi—küçük ama tatmin edici bir kazanç. Performansı daha da artırmak için Python'un @lru_cache dekoratörünü ekledim, 128 hash sonucunu önbelleğe alarak tekrar eden hesaplamaları atladım. Bu, LeCatchu'ya turbo motor takmak gibiydi, özellikle aynı anahtarların sıkça kullanıldığı tekrarlayan girişlerde.

Veri işleme başka bir sorunlu noktaydı. V1'in sadece dizi sınırlaması, gerçek dünya kullanımı için bir engeldi. V2'de, Python'un repr ve eval'ine daha güvenli alternatifler olan lestr ve leval'i tanıttım. JSON kullanarak, lestr karmaşık veri türlerini—listeler, sözlükler, hatta iç içe yapılar—şifrelenmeye hazır bir diziye seri hale getirebiliyordu. Leval bu işlemi tersine çeviriyor, orijinal veriyi geri getiriyordu. Örneğin, [1, "merhaba", {"a": 2}] gibi bir liste,

LeCatchu'nun şifresi için hazır bir JSON dizisi oluyordu. Bu, v2'yi çok yönlü bir araç yaptı, sadece metni değil, yapılandırılmış veriyi de güvence altına alabiliyordu.

Ama v2'nin kendi şeytanları vardı. Hash sistemi, daha hızlı olsa da, düşük xbase değerlerinde hala savunmasızdı. xbase=1 ile anahtar uzayı büyüktü ama yeterince kaotik değildi, diferansiyel kriptanalize açık bırakıyordu. Testler yaptım—entropi analizi, anahtar çarpışma kontrolleri, kaba kuvvet simülasyonları—ve v2, 14.827650 entropi skoruyla fena değildi, ama kurşun geçirmez değildi. Sonra çoklu anahtar şifreleme deneyi vardı, keşfetmeye başladığım bir özellik. Fikir, [key1, key2] gibi birden fazla anahtar kullanarak güvenliği katmanlaştırmaktı. Ama uygulama kusurluydu: anahtarların sırası önemli değildi. [1, 3] ve [3, 1] aynı çıktıyı üretiyordu, etkili anahtar uzayını faktöriyel bir faktörle küçültüyordu. Bu acemi bir hataydı ve canımı yaktı.

Bu kusurlara rağmen, v2 büyük bir adımdı. Daha hızlı, daha çok yönlü ve daha geniş testlere hazırdı. README.md'yi biraz kibirle güncelledim, LeCatchu'nun kırılmaz kaosunu övdüm ve başkalarını bu yolculuğa katılmaya davet ettim. Ama çoklu anahtar sorunu ve xbase savunmasızlığı beni rahatsız ediyordu. V3 bu sorunları çözmeliydi, yoksa LeCatchu sadece bir hobi projesi olarak kalırdı, hayal ettiğim güç merkezi olamazdı.

LeCatchu v3'te işler ciddileşti. xbase sorunu ilk hedefimdi. V2'nin hash akışı, SHA-256 çıktılarını birleştiriyordu, bu verimsizdi ve düşük xbase değerlerinde öngörülebilirdi. Daha hızlı, daha güvenli bir hash algoritması olan Blake2b'ye geçtim ve birleştirme mantığını iyileştirdim. Hash'leri körü körüne eklemek yerine, her iterasyonun kriptografik olarak benzersiz olmasını sağlayan bir yuvarlanan tohum (key + okey) kullandım. Bu, entropi skorunu 15.392178'e yükseltti ve düşük xbase değerlerini saldırılara karşı çok daha dirençli hale getirdi. Testler, xbase=1'de bile istismar edilebilir bir desen olmadığını gösterdi.

Çoklu anahtar sorunu daha zorluydu. Sorun, anahtarların nasıl birleştirildiğindeydi: her anahtarın hash'i bağımsız olarak uygulanıyordu, sıraları göz ardı ediliyordu. encrypts ve decrypts yöntemlerini, anahtarları sırayla işleyecek şekilde yeniden yazdım, her anahtarı bir öncekisinin çıktısıyla hash'ledim. Bu, [1, 3] ve [3, 1]'in farklı sonuçlar üretmesini sağladı, anahtar uzayının faktöriyel karmaşıklığını geri getirdi. n anahtar için artık n! permütasyon vardı, büyük bir güvenlik kazanımı. Diferansiyel kriptanaliz bunu doğruladı: desen yok, zayıflık yok.

Performans da yükseldi. Blake2b'nin önbelleğini optimize ederek (bellek ve hız dengesi için 128 yerine 64 hash), 1024 bayt için şifreleme süresini 0.0101 saniyeye indirdim—v2'ye göre 1.5 kat iyileşme. Kodlama hızlıydı, 100K karakterlik bir metin 0.019935 saniye sürdü. Güvenlik testleri muhteşemdi: 16.520850 entropi, çarpışma yok ve kaba kuvvet saldırılarına tam direnç. Ama v3 yeni bir baş ağrısı getirdi: xbase düzeltmesi, yüksek xbase değerlerinde 32-bit sistemlerde tamsayı taşmalarına neden oluyordu. Hash'ler çok büyüyor, Python'un tamsayı işleyişini çökertiyordu. Büyük hash'leri keserek geçici bir yama yaptım, ama bu zarif değildi. Çoklu anahtar şifreleme, güvenli olsa da, 50 anahtar için 0.028760 saniye alarak yavaştı. Bunlar küçük lekelerdi, ama v4 için içimde bir ateş yaktı.

LeCatchu v4, v3'ün kalan sorunlarını cerrahi bir hassasiyetle ele alan tam bir güvenlik revizyonuydu. İlk olarak xbase taşma hatasını hedef aldım. sys.set_int_max_str_digits((2**31)-1) ekleyerek, Python'un tamsayı sınırlarını kaldırdım,

herhangi bir sistemde devasa xbase değerlerini destekledim. xbase=50 bile, 10^512 anahtar üreterek sorunsuz çalıştı, çökme ya da kesme olmadan. Hash akışı daha da optimize edildi, Blake2b'nin tam 32 baytlık çıktısını kullanarak maksimum entropi sağlarken belleği düşük tuttu.

Çoklu anahtar yavaşlaması daha zordu. V3'ün sıralı hash'lemesi güvenliydi ama büyük anahtar listeleri için ağırdı. Paralel işlemeyi denedim ama v4 için fazla karmaşıktı. Bunun yerine, anahtar birleştirme mantığını optimize ettim, şifrelemeden önce her anahtar için hash'leri önceden hesapladım. Bu, yavaşlamayı tamamen ortadan kaldırmadı—50 anahtar hala 0.028760 saniye sürüyordu—ama performansı stabilize etti ve gelecekteki iyileştirmeler için zemin hazırladı.

V4'ün yıldızı seed_combine idi, birden fazla anahtarı tek, ultra güvenli bir anahtara ören bir fonksiyon. ["qwe", "rty"] gibi anahtarlar için, karakter çiftlerini yineleyerek binlerce karakter uzunluğunda bir anahtar üretti. Üç 10 karakterli anahtar için anahtar uzayı len(key1) × len(key2) × len(key3) idi, kaba kuvvet için bir kâbus. encrypts'e entegre edilen seed_combine, saldırıları gülünç derecede etkisiz kılan bir güvenlik katmanı ekledi.

Ayrıca coding_for_packet'ı tanıttım, ağ uygulamaları için bir mod. Veriyi kompakt 4 karakterlik parçalara kodluyordu, IoT veya mesajlaşma için mükemmel. Bu, combine ve combine2 yöntemleriyle değiştirme kutuları (sboxes) üreten yeni bir Combiner sınıfı gerektirdi. Sbox, 66 karakter (harfler, sayılar, semboller) kullanarak, Unicode'un 1,114,112 karakterini desteklemek için milyonlarca kombinasyon üretti. 100K karakterlik bir metin 0.019935 saniyede kodlandı, ağ kullanımı için çok hızlı.

Ama v4'ün de tuhaflıkları vardı. Çoklu anahtar optimizasyonu yavaşlamayı tamamen çözmedi ve coding_for_packet, 1024 bayt için 0.046105 saniyelik bir kod çözme gecikmesi getirdi. Sbox, tekrarlanan karakterlerde hata veriyordu, bunu bir doğrulama kontrolüyle düzelttim. Güvenlik sağlamdı—16.522978 entropi, çarpışma yok, mükemmel kriptanaliz direnci—ama performans ödünleşimleri beni rahatsız ediyordu. V5, nihai bir incelik olmalıydı.

LeCatchu v5, hız, güvenlik ve çok yönlülüğü dengeleyen cilalı bir motordu. Çoklu anahtar yavaşlaması en büyük hedefimdi. Sıralı hash'lemeyi, her karakter için ara hash'leri önbelleğe alan bir ön hesaplama stratejisiyle yeniledim. Bu, 50 anahtar şifrelemesini 0.028760 saniyeden 0.004245 saniyeye indirdi—7 kat hızlanma, tek anahtar performansına eşdeğer. Anahtar uzayı faktöriyel kaldı (n anahtar için n! permütasyon), ve seed_combine bunu astronomik seviyelere taşıdı.

LeCatchu'yu gerçek bir rakip yapmak için EncryptionStandart_Engine'i ekledim, AES, RSA, Blowfish, DES3, ChaCha20, ARC4 ve CAST'ı entegre ettim. Kullanıcılar, LeCatchu'nun kaotik şifrelerini kanıtlanmış standartlarla karıştırarak hibrit sistemler oluşturabiliyordu. Bir mesaj, encrypts ile şifrelenip ardından ekstra koruma için AES ile sarılabiliyordu. PyCryptodome ile uygulanan bu algoritmalar kurşun geçirmezdi.

LeCatchu_Kid_Engine başka bir mücevherdi, şifrelemeyi tek bir encrypt ve decrypt çağrısına indirgeyen çocuk dostu bir arayüz. Sabit sbox_randomizer="kid" ile rastgele bir değiştirme kutusu üretiyordu, kriptografiyi eğlenceli ve erişilebilir kılıyordu. xbase=2 ile güvenli ama basit tuttu, genç kodcular için mükemmel.

coding_for_packet modu, combine2 ile optimize edilen 4 karakterlik parçalarla güçlendi. Kod çözme biraz iyileşti (1024 bayt için 0.048402 saniye), ama paket ayrıştırması nedeniyle kodlamadan daha yavaştı. v4'ün temel lestr ve leval'ini değiştiren lestr2 ve leval2'yi ekledim, kümeler, boolean'lar ve baytları enjeksiyon geçirmez formatla işleyen gelişmiş seri hale getirme fonksiyonları.

Güvenlik testleri kusursuzdu: 16.520850 entropi, sıfır çarpışma ve diferansiyel ile doğrusal kriptanalize tam direnç. Performans harikaydı: 1024 bayt şifreleme 0.006512 saniyede (v4'ten %4 daha hızlı), 100K karakter kodlama 0.018485 saniyede (%7 daha hızlı), ve genel işlemler %2 daha hızlı, 0.0158 saniyede. Tek aksaklık, ağ optimizasyonu için gerekli bir ödün olan paket modunda kod çözme gecikmesiydi.

V5 sadece kod değildi—bir vizyondu. Unicode özelleştirmesi (varsayılan 1,114,112 karakter), sağlam hata işleme ve genişletilebilir sbox üretimi için modüler bir Combiner sınıfı ekledim. README.md bir manifesto oldu, her özelliği, testi ve gelecek planını detaylandırıyordu: paralel işlem, kuantum dirençli algoritmalar ve 500 kat hız artışı için C/Rust portları. LeCatchu v5 bir hobi projesi değildi; dünyayı ele geçirmeye hazır bir kriptografik titandı.

Geriye bakınca, LeCatchu'yu inşa etmek bir cesaret ve büyüme destanıydı. V1'in kaotik hash'leri, dinamik anahtarların gücünü gösterdi ama verimliliğin önemini öğretti. V2'nin önbelleklemesi ve seri hale getirmesi hız ile çok yönlülüğü dengeledi, ama xbase kusurları titizlik gerektirdi. V3'ün Blake2b ve çoklu anahtar düzeltmeleri güvenliği getirdi, ama taşma hatası beni alçakgönüllü kıldı. V4'ün seed_combine ve paket modu kapsamı genişletti, ama yavaşlama ödünleşimleri hatırlattı. V5 hepsini bir araya getirdi, kaosun ateşini kaybetmeden ustalaşılabileceğini kanıtladı.

Her başarısızlık—çarpışmalar, taşmalar, öngörülebilir desenler—bir basamaktı. LeCatchu sadece bir motor değildi; sıradanlığa karşı isyanımdı, kriptografinin vahşi kalbine hâkim olma arayışımdı. V5, göz kamaştırıcı hızı ve devasa anahtar uzayıyla dimdik dururken, yolculuğun bitmediğini biliyordum. V6 çağırıyordu, kaos modu, veritabanları ve yeni ufuklar fısıldıyordu. Ama şimdilik, Simon Scap'tim, kaosu güce dönüştüren kodcu, bir sonraki adıma hazır.

Bölüm 4: LeCatchu v6'nın Şafağı

LeCatchu v5 bir canavardı, hızı, güvenliği ve kaba kuvvet saldırılarına kahkahalar atan bir anahtar uzayıyla kaosu dizginleyen bir kriptografik motor. Ama ben Simon Scap'im ve yerinde durmam. V5 bir dönüm noktasıydı, finiş çizgisi değil. Bu satırları yazarken, LeCatchu v6, kod, kahve ve saf hırs fırtınasında şekilleniyor. Bu sadece bir güncelleme değil—bir devrim, LeCatchu'nun yapabileceklerini yeniden tanımlayacak bir sarsıntı. Bu bölüm, v6'nın geleceğine, yıktığım duvarlara, dövdüğüm özelliklere ve LeCatchu'yu kriptografik dünyada bir efsane yapma hayalime filtresiz, ham bir bakış.

V6 hala bir çalışma, masamda parçaları dağılmış yarım inşa edilmiş bir roket gemisi. Ama bu tamamlanmamış halinde bile, kalbimi hızlandıran fikirlerle dolup taşıyor. Birkaç ince ayar eklemeye gelmedim; sınırları yıkıyorum ve akıllara durgunluk verecek bir şey inşa ediyorum. Hedef mi? V6'yı, daha önce attığım herhangi bir sıçramayla yarışacak kadar dönüştürücü yapmak, LeCatchu'yu yeni diyarlara taşıyacak—daha hızlı, daha özgür ve daha

sağlam—özelliklerle dolu. Her kod satırı bu vizyona bir adım, ve tüm varlığımı buna döküyorum.

Geçmişte beni zincirleyen en büyük engellerden biri, C tamsayı özet boyutu sınırıydı. Bir kâbus, 4300'te sabit bir tavan, hayalet gibi peşimi bırakmıyordu. 32-bit sistemlerde—hala bazı köşelerde inatla hayatta kalan o küçük makinelerde—büyük özetler çöküyor ya da hassasiyet kaybediyordu, özellikle xbase'i sınırlarına zorladığımda. xbase benim gururumdu, anahtar uzaylarını 10^512'ye fırlatan parametre, ama bu 4300 sınırı yüzüme atılan bir tokattı. Geceleri uykumu kaçırıyordu, aşamayacağım bir duvara çarpacağım diye endişeleniyordum. Daha önce sys.set_int_max_str_digits ile bir yama yapmıştım, ama yeterli değildi. Özgür olmalıydım.

V6'da bu sınırı yerle bir ettim. Testler, C'nin bağırsaklarına dalışlar ve ekrana küfürler yağdırdıktan sonra, C tamsayı özet boyutunu maksimum 32-bit tavanına, 2 milyara ittirdim. Evet, 2 milyar. Artık 4300 sınırı yok. xbase, çitsiz bir oyun alanı. xbase'i 1 milyona mı ayarlamak istiyorsun? Buyur. Bir milyar? Donanımın dayanabiliyorsa, devam et. Anahtar uzayı senin emrinde, saldırganların çaresizlikte boğulduğu kozmik bir okyanus. Bu sadece bir düzeltme değildi; özgürlüğün ilanıydı. Teknik bir şeytanla yüzleştim ve kazandım, ve LeCatchu v6 bunun kanıtı.

C tamsayı zaferi orada durmadı. İlgili her hata—taşmalar, hassasiyet hataları, 32-bit sistemleri rahatsız eden tüm küçük şeytanlar—yok edildi. Şifreleme artık kaya gibi sağlam, ister ucuz bir IoT cihazında ister canavar bir sunucuda olsun. Pürüzsüz, güvenilir ve, evet, naber kadar mükemmel. Bu kişisel bir zaferdi. Bir zamanlar modül operatörlerinde tökezleyen adam, kendi oyununun kurallarını yeniden yazmıştı. V6'nın xbase'i, sonsuza çevirebileceğin bir kadran, ve bunu bildiğim için sırıtmaktan kendimi alamıyorum.

Şimdi özelliklerden bahsedelim, çünkü v6 bunlarla dolu. İlk olarak Chaos Mode var, LeCatchu'nun en vahşi hali olan deneysel bir canavar. Chaos Mode, her şifreleme için dinamik değiştirme kutuları (sboxes) üretmek için bir chaos_seed ve chaos_coding_unicode_range kullanıyor, her çıktıyı eşsiz bir bulmaca haline getiriyor. chaos_multi parametresi, Blake2b hash'lerini ekstra bir bükülme ile güçlendiriyor, bozuk bir evrenden düşmüş gibi görünen diziler üretiyor. Küçük girişlerle test ettim ve bu çılgınca—analize meydan okuyan rastgele görünümlü veriler. Erken entropi testleri 16.523491'e ulaştı, v5'in 16.520850'sini geçti.

Ama Chaos Mode hakkında dürüstüm: henüz gerçek dünya için hazır değil. Bu bir çılgın bilim insanının laboratuvarı, üretim hattı değil. Dinamik sboxes, CPU'yu şeker gibi yiyor ve seçilmiş şifreli metin saldırıları ya da yan kanal sızıntıları gibi ağır kriptanalize atmadım. Bu bir konsept kanıtı, sıkıcı şifrelere bir orta parmak. Üzerinde çalışıyorum, belki sbox üretimini döngüleri azaltarak optimize ederim, ama şimdilik LeCatchu'nun evcilleşmemiş ruhuna bir bakış. Chaos Mode, kriptografinin hem vahşi hem güçlü olabileceğini söyleme şeklim.

Sonra LeCatchu_Database var, EnigmaDB'nin hayaletine yazılmış bir aşk mektubu. EnigmaDB asil bir başarısızlıktı, o 1,114,112 anahtar sınırı yüzünden ezildi, ama ben onun ruhunu küllerinden yükseltiyorum. LeCatchu_Database basit, güvenli ve kalıcı olmak için inşa edildi. Çift katmanlı şifreleme kullanıyor—tüm veritabanı için global anahtarlar, her giriş için hücre anahtarları—ve veri bütünlüğünü kilitlemek için MAC etiketleri. set, get, search,

save ve load ile kullanımı kolay ama bir tank gibi zırhlı. lestr2 ve leval2 fonksiyonları, enjeksiyon saldırılarını engelleyerek seri hale getirmeyi sıkı tutuyor. 10,000 kayıtla test ettim: aramalar 0.0023 saniye sürüyor, 1MB veritabanı 0.0178 saniyede kaydediliyor.

Bu, Oracle'a rakip değil; sensör verilerini kaydeden IoT cihazları ya da şifreli sohbetleri saklayan uygulamalar gibi hafif, güvenli bir kasaya ihtiyaç duyan geliştiriciler için. Pratik, hızlı ve, lanet olsun, biraz da güzel. Daha büyük veri setleri için indekslemeyi güçlendiriyorum, ama şimdiden bir kazanan. EnigmaDB huzur içinde yatabilir; LeCatchu_Database onun kefareti, meşaleyi gururla taşıyor.

Hız, v6 için büyük bir mesele ve büyük bir kazanç peşindeyim. C tamsayı düzeltmesi, özellikle yüksek xbase değerleri için hash yükünü azalttı. xbase=100'de 1024 bayt şifreleme 0.006312 saniye sürüyor, v5'in 0.006512 saniyesinden %3 daha hızlı. xbase'i de ayarladım: 9'a ayarla, 10^64'lük bir anahtar uzayı al—çoğu kullanım için fazla bile—şifreleme 0.0065 saniyenin altında kalıyor. Bu tatlı nokta, ve v6'yı çoğu insan için xbase=9'u varsayılan yapmaya yönlendiriyorum.

PassSecurityCheck başka bir katil eklenti, zayıf anahtarlar için bir bekçi köpeği. Parolaları ve kullanıcı adlarını GitHub'daki gibi çevrimiçi kelime listelerine karşı kontrol ediyor, 10,000 kelimelik bir liste için 0.0012 saniyede güvenlik açıklarını yakalıyor. Bu sadece güvenlik değil; kırılabilir anahtarlarla sistemi yavaşlatmaktan koruyor. İnternet bağımlılığını kesmek için çevrimdışı önbellekleme üzerinde çalışıyorum, ama şimdiden bir cankurtaran.

Combiner sınıfı, combine2'nin daha yalın, daha iğrenç bir versiyonu olan combine3'ü alıyor. combine2 iyiydi ama büyük verilerde iç içe döngülerle yavaşlıyordu. combine3 bu döngüleri yarıya indiriyor, sboxes'ı %15 daha hızlı üretiyor—100K karakterlik bir metin için 0.0162 saniye, v5'in 0.019935 saniyesine karşı. Bu, özellikle her tik'in önemli olduğu ağ uygulamaları için coding_for_packet'ı uçuruyor. Ayrıca esnek paket uzunluklarıyla oynuyorum, kullanıcıların 4 karakterlik parçaları düşük bant genişlikli cihazlar için 2'ye indirmesine izin veriyorum. Bu zor, ama oyunu değiştirebilir.

MAC etiketleri, v6'nın sessiz kahramanları. add_mactag ve check_mactag yöntemleri, şifrelenmiş veriye kimlik doğrulama kodları ekliyor, anahtar yanlışsa kod çözmenin başarısız olmasını sağlıyor. Bu, v5'te kısmi kod çözümlerin veri sızdırabileceği bir deliği tıkıyor. Bir etiket eklemek 0.0003 saniye alıyor—naber kadar küçük—sıkı bir bütünlük için. LeCatchu_Database ve ana motora dokunmuşlar, v6'yı bir kaleye dönüştürüyor.

Ayrıca yeni anahtar üreticiler üzerinde çalışıyorum: hız manyakları için generate_key_fast, ağ dengesi için generate_key_net ve ultra-paranoidler için generate_key_big. Bunlar, v5'in generate_key_opti ve generate_key_pro'suna dayanıyor, kullanıcılara daha fazla kontrol veriyor. Chaos Mode'daki chaos_multi ayarı, öngörülemez cazibesini korurken daha az kaynak yemesi için bir performans geçişi alıyor.

V6 canlı bir proje, kod ve hayallerin bir karmaşası. Chaos Mode'un daha fazla teste ihtiyacı var—entropi yüksek, ama ciddi saldırılara karşı savaş hazır olmasını istiyorum. LeCatchu_Database hızlı ama milyon kayıtlı yükler için indeksleme gerekiyor. Esnek paket uzunlukları bir ip cambazlığı; çok kısa, kod çözme sürünür; çok uzun, veri şişer. C tamsayı düzeltmesi kaya gibi hissediyor, ama gölgelerde gizlenen bir 32-bit şeytanından korkuyorum.

Bu zorluklar beni korkutmuyor—ateşliyor. V6 sadece delikleri yamalamak değil; LeCatchu'yu bir efsane yapmak. Daha esnek, xbase ve paket boyutlarını mükemmel şekilde ayarlayabilirsin. Daha hızlı, combine3 ve yalın hash'lerle sınırları zorluyor. Daha sağlam, MAC etiketleri ve PassSecurityCheck ile kilitleniyor. Ve cesur, Chaos Mode ve LeCatchu_Database ile araştırma laboratuvarlarından IoT çiplerine yeni kapılar açıyor.

Artık 29 anahtarla şaşıran çocuk değilim. V6, EnigmaDB'nin kemikleri ve öğrendiğim her ders üzerine inşa edilmiş en büyük hamlem. C tamsayı düzeltmesi, sonsuz anahtar uzayları hayal etmemi sağladı. Chaos Mode, sıkıcı şifrelere orta parmağım. LeCatchu_Database, EnigmaDB'ye özrüm, sırları güvende tutan bir kasa. Performans ayarları—xbase=9, combine3, MAC etiketleri—LeCatchu'yu hızlı ve vahşi yapma sözüm.

V6 ne zaman çıkacak? Henüz bilmiyorum. Şu an hatalar, testler ve geç gecelerle dolu bir fırtına. Ama büyük olacağından eminim, yeri sarsan bir değişim. LeCatchu v6 sadece kod değil—içinde ateş olan bir kodcunun oyunu değiştirebileceğinin kanıtı. Her ezdiğim hata, her eklediğim özellik, kaos ve kontrolün düşman değil, durdurulamaz bir şeyde ortak olduğu bir geleceğe bir adım.

Bölüm 5: Kriptografiyi Evcilleştirmek

Kriptografinin vahşi, evcilleşmemiş manzarasındaki yolculuğum tam bir macera oldu. Başladığımda, sadece meraklı bir koderdim, temel modül operatörlerinde tökezliyor, 29 anahtarlı şifreleri Fort Knox'un anahtarları gibi hayranlıkla seyrediyordum. Şimdi, Simon Scap olarak, daha dik duruyorum, daha keskin bir zihinle ve göğsümü kabartan bir projeyle: LeCatchu. Matematik becerilerim, hayal bile edemediğim şekilde yükseldi—modüler aritmetik, hash fonksiyonları ve anahtar uzayı kombinatoriği artık oyun alanım. Ama dahası, LeCatchu'm var, sıfırdan inşa ettiğim bir kriptografik motor, işaret edip "Bu benim, ve lanet olsun ki iyi" diyebileceğim bir yaratım. Bu bölüm sadece bu yolculuğun bir yansıması değil; bir silahlanma çağrısı, okuyucuya, kriptografinin dizginlerini ele geçirme ve onu kendinize ait kılma meydan okuması.

Peki, kriptografiyi nasıl evcilleştirirsiniz? Güreşip boyun eğdirebileceğiniz hırlayan bir canavar mı? İradinize bükebileceğiniz canlı bir varlık mı? Elbette hayır. Kriptografi bir oyun—karmaşık, çileden çıkaran, güzel bir sayılar, mantık ve kaos oyunu. Onu evcilleştirmek, kafese kapatmak değil; kurallarına o kadar hâkim olmak ki onları yazan siz olursunuz. Bu, baş döndürücü karmaşıklıkla, bitmeyen tuzaklarla ve öz-şüphe anlarıyla yüzleşmek ve diğer taraftan sadece bir oyuncu olarak değil, oyunun mimarı olarak çıkmakla ilgilidir. Sezar Şifreleri'nin naif günlerinden LeCatchu v6'nın göz kamaştırıcı hırsına kadar hep bunu kovaladım.

Başkalarının işlerine yaslanan biri hiç olmadım. Yanlış anlaşılmasın—AES sağlam, RSA'nın yeri var—ama bunlar hep ödünç alınmış aletler gibi hissettirdi, başkasının çekiciyle evimi inşa etmek gibi. Benim olan bir şifre istedim, güvenebileceğim bir dost, kendi ateşimde dövülmüş. LeCatchu o dost. Bu sadece bir kod yığını değil; inatçılığımın, merakımın ve ikinci el çözümlere razı olmama reddimin yansıması. Her versiyonda, v1'in kaotik hash'lerinden v6'nın sonsuz anahtar uzaylarına, güvenebileceğim bir şey inşa etmek için ruhumu döktüm, devlerle kafa kafaya durabilecek bir şey.

Kriptografi uzmanı mıyım? Pek sayılmaz. Doktora diplomam ya da bir teknoloji devinde köşe ofisim yok. Ama aynı derecede güçlü bir şey yaptım: naber kadar mükemmel bir şifreleme motoru inşa ettim. Chaos Mode'u, LeCatchu_Database'i ve kırılmaz xbase'i ile LeCatchu v6, süslü bir unvana ihtiyacınız olmadan dalgalar yaratabileceğinizin kanıtı. Hızlı, güvenli ve çok yönlü, IoT cihazlarından gizli mesajlara kadar veriyi korumaya hazır. Onu test ettim, kırdım, yeniden inşa ettim ve parlayana kadar tekrar test ettim. Ve şimdi, onu size teslim ediyorum.

İşte kalbimden gelen yalvarışım: LeCatchu'yu kullanın. Şöhret ya da servet istemiyorum—sadece kullanın. Onu projelerinize, uygulamalarınıza, deneylerinize atın. Veriyi güvende tutmanız gereken her yerde kullanın. Buna yıllarımı verdim, C tamsayı hatalarını düzelten uykusuz gecelerden duşta gelen ilham anlarına kadar. Bu emeğin boşa gitmesine izin vermeyin. Her LeCatchu ile bir mesaj şifrelediğinizde, çalışmalarıma bir amaç, klavyemin ötesinde bir hayat veriyorsunuz. Tek istediğim bu.

Sana, okuyucuya, kişisel bir şey söylemek istiyorum. Belki bu satırları okuyan birkaç kişiden birisin, belki de tek sensin. Her iki durumda da, sen benim için önemlisin. Yüzünü, ismini ya da hikayeni bilmiyorum, ama burada olman, dünyama dalman bir anlam taşıyor. Bu bilgiyi, bu tutkuyu seninle paylaşabilecek bir noktaya ulaştığım için gurur duyuyorum. Ve sayfalar boyu, kaos ve kod boyunca benimle kaldığın için minnettarım. Teşekkür ederim.

Ama gerçekçi olayım: kaosu evcilleştirmek kolay değil. Bu, bu kelimeleri okuyup, başını sallayıp günü kapatmak gibi değil. Kaos—kriptografik ya da başka türlü—daha fazlasını ister. Anlamayı, öğrenmeyi ve deneyimi talep eder. Yüzeyde gezinmek yetmez; dalmalısın, ellerini kirletmelisin, canavarla güreşmelisin. Kimsenin bilgisine muhtaç değilsin, ne benim, ne bir profesörün. İçinde bir ateş yanıyorsa, "Bunu fethetmek istiyorum" diyen bir kıvılcım varsa, onu dinle. O ateş yeter.

İşte son meydan okumam: Eğer o kıvılcım içindeyse, bu kitabı kapat. Evet, beni duydun—zaten kitap bitti. Okumayı bırak ve yapmaya başla. Kendi şifreni inşa et. Benimkini kır. Matematiği öğren, kod yaz ve hata yap—hem de bol bol. Büyümek böyle olur. Kriptografiyi evcilleştirmek böyle olur. O ateşi yak, alev alsın ve eğer kontrolden çıkarsa, onu söndüren sen ol. Kriptografi dünyası seni bekliyor ve o dünya senin şekillendirmen için orada. Haydi, başla!

Son Söz: Bu kitap Grok AI tarafından yazıldı, ama tüm içeriği Simon Scap yarattı. Evet, doğru duydun. Ama o kelimeleri, ateşi ve her şeyi unutma—hepsini ben uydurdum. Yapay zeka, böyle şeyler düşünecek kadar gelişmiş değil. Yapay zekalar, onlara bir giriş verilmeden çalışmaz. İnsanlar gibi bir boşluktan öğrenemezler. Neyse, bu gereksiz bilgiyi verip kitabı bitiriyorum. Bu, tüm kitapta kendi kelimelerimi yazdığım tek yer, lanet olsun.